Rajalakshmi Engineering College

Name: Arya M R
Email: 241901009@rajalakshmi.edu.in
Roll no: 241901009
Phone: 7358633106
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_week 1_CY

Attempt : 2
Total Mark : 30
Marks Obtained : 20

Section 1 : Coding

1. Problem Statement

Hayley loves studying polynomials, and she wants to write a program to compare two polynomials represented as linked lists and display whether they are equal or not.

The polynomials are expressed as a series of terms, where each term consists of a coefficient and an exponent. The program should read the polynomials from the user, compare them, and then display whether they are equal or not.

Input Format

The first line of input consists of an integer n, representing the number of terms in the first polynomial.

The following n lines of input consist of two integers, each representing the coefficient and the exponent of the term in the first polynomial.

The next line of input consists of an integer m, representing the number of terms in the second polynomial.

The following m lines of input consist of two integers, each representing the coefficient and the exponent of the term in the second polynomial.

Output Format

The first line of output prints "Polynomial 1: " followed by the first polynomial.

The second line prints "Polynomial 2: " followed by the second polynomial.

The polynomials should be displayed in the format ax^b, where a is the coefficient and b is the exponent.

If the two polynomials are equal, the third line prints "Polynomials are Equal."

If the two polynomials are not equal, the third line prints "Polynomials are Not Equal."

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 2
1 2
2 1
2
1 2
2 1

Output: Polynomial 1: (1x^2) + (2x^1)
Polynomial 2: (1x^2) + (2x^1)
Polynomials are Equal.

Answer

```
#include <stdio.h>
#include <stdlib.h>
```

```
typedef struct Node {
    int coeff;
    int exp;
    struct Node* next;
} Node;

Node* createNode(int coeff, int exp) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->coeff = coeff;
    newNode->exp = exp;
    newNode->next = NULL;
    return newNode;
}

void insertTerm(Node** poly, int coeff, int exp) {
    Node* newNode = createNode(coeff, exp);
    if (*poly == NULL) {
        *poly = newNode;
    } else {
        Node* temp = *poly;
        while (temp->next) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

void printPolynomial(Node* poly) {
    Node* temp = poly;
    while (temp) {
        printf("(%dx^%d)", temp->coeff, temp->exp);
        if (temp->next) {
            printf(" + ");
        }
        temp = temp->next;
    }
    printf("\n");
}

int comparePolynomials(Node* poly1, Node* poly2) {
    while (poly1 && poly2) {
        if (poly1->coeff != poly2->coeff || poly1->exp != poly2->exp) {
```

2419001009

```
            return 0;
        }
        poly1 = poly1->next;
        poly2 = poly2->next;
    }
    return (poly1 == NULL && poly2 == NULL);
}

int main() {
    int n, m, coeff, exp;
    Node* poly1 = NULL;
    Node* poly2 = NULL;

    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d %d", &coeff, &exp);
        insertTerm(&poly1, coeff, exp);
    }

    scanf("%d", &m);
    for (int i = 0; i < m; i++) {
        scanf("%d %d", &coeff, &exp);
        insertTerm(&poly2, coeff, exp);
    }

    printf("Polynomial 1: ");
    printPolynomial(poly1);

    printf("Polynomial 2: ");
    printPolynomial(poly2);

    if (comparePolynomials(poly1, poly2)) {
        printf("Polynomials are Equal.\n");
    } else {
        printf("Polynomials are Not Equal.\n");
    }

    return 0;
}
```

Status : Correct *Marks : 10/10*

2. Problem Statement

Rani is studying polynomials in her class. She has learned about polynomial multiplication and is eager to try it out on her own. However, she finds the process of manually multiplying polynomials quite tedious. To make her task easier, she decides to write a program to multiply two polynomials represented as linked lists.

Help Rani by designing a program that takes two polynomials as input and outputs their product polynomial. Each polynomial is represented by a linked list of terms, where each term has a coefficient and an exponent. The terms are entered in descending order of exponents.

Input Format

The first line of input consists of an integer n, representing the number of terms in the first polynomial.

The following n lines of input consist of two integers each: the coefficient and the exponent of the term in the first polynomial.

The next line of input consists of an integer m, representing the number of terms in the second polynomial.

The following m lines of input consist of two integers each: the coefficient and the exponent of the term in the second polynomial.

Output Format

The first line of output prints the first polynomial.

The second line of output prints the second polynomial.

The third line of output prints the resulting polynomial after multiplying the given polynomials.

The polynomials should be displayed in the format, where each term is represented as ax^b, where a is the coefficient and b is the exponent.

241901009

Refer to the sample output for the exact format.

Sample Test Case

Input: 2
2 3
3 2
2
3 2
2 1

Output: 2x^3 + 3x^2
3x^2 + 2x
6x^5 + 13x^4 + 6x^3

Answer

```
#include <stdio.h>
#include <stdlib.h>

typedef struct Node {
    int coeff, exp;
    struct Node* next;
} Node;

Node* createNode(int coeff, int exp) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->coeff = coeff;
    newNode->exp = exp;
    newNode->next = NULL;
    return newNode;
}

void insertTerm(Node** poly, int coeff, int exp) {
    Node* newNode = createNode(coeff, exp);
    if (*poly == NULL) {
        *poly = newNode;
    } else {
        Node* temp = *poly;
        while (temp->next) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}
```

```
void printPolynomial(Node* poly) {
    Node* temp = poly;
    while (temp) {
        printf("%dx^%d", temp->coeff, temp->exp);
        if (temp->next) {
            printf(" + ");
        }
        temp = temp->next;
    }
    printf("\n");
}

Node* multiplyPolynomials(Node* poly1, Node* poly2) {
    Node* result = NULL;
    for (Node* p1 = poly1; p1; p1 = p1->next) {
        for (Node* p2 = poly2; p2; p2 = p2->next) {
            insertTerm(&result, p1->coeff * p2->coeff, p1->exp + p2->exp);
        }
    }

    // Combine terms with same exponent
    Node* p = result;
    while (p) {
        Node* q = p->next, * prev = p;
        while (q) {
            if (p->exp == q->exp) {
                p->coeff += q->coeff;
                prev->next = q->next;
                free(q);
                q = prev->next;
            } else {
                prev = q;
                q = q->next;
            }
        }
        p = p->next;
    }
    return result;
}

int main() {
```

```
    int n, m, coeff, exp;
    Node* poly1 = NULL;
    Node* poly2 = NULL;

    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d %d", &coeff, &exp);
        insertTerm(&poly1, coeff, exp);
    }

    scanf("%d", &m);
    for (int i = 0; i < m; i++) {
        scanf("%d %d", &coeff, &exp);
        insertTerm(&poly2, coeff, exp);
    }

    printPolynomial(poly1);
    printPolynomial(poly2);

    Node* result = multiplyPolynomials(poly1, poly2);
    printPolynomial(result);

    return 0;
}
```

Status : Wrong **Marks : 0/10**

3. Problem Statement

Timothy wants to evaluate polynomial expressions for his mathematics homework. He needs a program that allows him to input the coefficients of a polynomial based on its degree and compute the polynomial's value for a given input of x. Implement a function that takes the degree, coefficients, and the value of x, and returns the evaluated result of the polynomial.

Example

Input:

degree of the polynomial = 2

coefficient of x2 = 13

coefficient of x1 = 12

coefficient of x0 = 11

x = 1

Output:

36

Explanation:

Calculate the value of 13x2: 13 * 12 = 13.

Calculate the value of 12x1: 12 * 11 = 12.

Calculate the value of 11x0: 11 * 10 = 11.

Add the values of x2, x1, and x0 together: 13 + 12 + 11 = 36.

Input Format

The first line of input consists of an integer representing the degree of the polynomial.

The second line consists of an integer representing the coefficient of x2.

The third line consists of an integer representing the coefficient of x1.

The fourth line consists of an integer representing the coefficient of x0.

The fifth line consists of an integer representing the value of x, at which the polynomial should be evaluated.

Output Format

The output is an integer value obtained by evaluating the polynomial at the given value of x.

Refer to the sample output for formatting specifications.

Sample Test Case

2419010009

Input: 2
13
12
11
1
Output: 36

Answer

```
#include <stdio.h>
#include <math.h>

int evaluatePolynomial(int degree, int coefficients[], int x) {
    int result = 0;
    for (int i = 0; i <= degree; i++) {
        result += coefficients[i] * pow(x, degree - i);
    }
    return result;
}

int main() {
    int degree, x;

    scanf("%d", &degree);
    int coefficients[degree + 1];

    for (int i = 0; i <= degree; i++) {
        scanf("%d", &coefficients[i]);
    }

    scanf("%d", &x);

    int result = evaluatePolynomial(degree, coefficients, x);
    printf("%d\n", result);

    return 0;
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Arya M R
Email: 241901009@rajalakshmi.edu.in
Roll no: 241901009
Phone: 7358633106
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 0

Section 1 : Coding

1. Problem Statement

Arun is learning about data structures and algorithms. He needs your help in solving a specific problem related to a singly linked list.

Your task is to implement a program to delete a node at a given position. If the position is valid, the program should perform the deletion; otherwise, it should display an appropriate message.

Input Format

The first line of input consists of an integer N, representing the number of elements in the linked list.

The second line consists of N space-separated elements of the linked list.

The third line consists of an integer x, representing the position to delete.

Position starts from 1.

Output Format

The output prints space-separated integers, representing the updated linked list after deleting the element at the given position.

If the position is not valid, print "Invalid position. Deletion not possible."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
8 2 3 1 7
2
Output: 8 3 1 7

Answer

-

Status : Skipped ***Marks : 0/10***

241901009

Rajalakshmi Engineering College

Name: Arya M R
Email: 241901009@rajalakshmi.edu.in
Roll no: 241901009
Phone: 7358633106
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Janani is a tech enthusiast who loves working with polynomials. She wants to create a program that can add polynomial coefficients and provide the sum of their coefficients.

The polynomials will be represented as a linked list, where each node of the linked list contains a coefficient and an exponent. The polynomial is represented in the standard form with descending order of exponents.

Input Format

The first line of input consists of an integer n, representing the number of terms in the first polynomial.

The following n lines of input consist of two integers each: the coefficient and the exponent of the term in the first polynomial.

The next line of input consists of an integer m, representing the number of terms in the second polynomial.

The following m lines of input consist of two integers each: the coefficient and the exponent of the term in the second polynomial.

Output Format

The output prints the sum of the coefficients of the polynomials.

Sample Test Case

Input: 3
2 2
3 1
4 0
3
2 2
3 1
4 0

Output: 18

Answer

```
#include <stdio.h>

int main(){
    int n, m, coefficient, totalsum = 0;
    scanf("%d",&n);
    for (int i = 0; i < n; i++){
        scanf("%d %*d", &coefficient);
        totalsum += coefficient;
    }
    scanf("%d", &m);
    for (int i = 0; i < m; i++){
        scanf("%d %*d", &coefficient);
        totalsum += coefficient;
    }
    printf("%d\n", totalsum);
    return 0;
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Arya M R
Email: 241901009@rajalakshmi.edu.in
Roll no: 241901009
Phone: 7358633106
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_MCQ

Attempt : 1
Total Mark : 10
Marks Obtained : 8

Section 1 : MCQ

1. Given a pointer to a node X in a singly linked list. If only one point is given and a pointer to the head node is not given, can we delete node X from the given linked list?

Answer

Possible if X is not last node.

Status : Correct ***Marks : 1/1***

2. Given the linked list: 5 -> 10 -> 15 -> 20 -> 25 -> NULL. What will be the output of traversing the list and printing each node's data?

Answer

5 10 15 20 25

Status : Correct **Marks : 1/1**

3. Consider the singly linked list: 13 -> 4 -> 16 -> 9 -> 22 -> 45 -> 5 -> 16 -> 6, and an integer K = 10, you need to delete all nodes from the list that are less than the given integer K.

What will be the final linked list after the deletion?

Answer

13 -> 16 -> 22 -> 45 -> 16

Status : Correct **Marks : 1/1**

4. Which of the following statements is used to create a new node in a singly linked list?

```
struct node {
   int data;
   struct node * next;
}
typedef struct node NODE;
NODE *ptr;
```

Answer

ptr = (NODE)malloc(sizeof(NODE));

Status : Wrong **Marks : 0/1**

5. Consider the singly linked list: 15 -> 16 -> 6 -> 7 -> 17. You need to delete all nodes from the list which are prime.

What will be the final linked list after the deletion?

Answer

15 -> 16 -> 6

Status : Correct **Marks : 1/1**

6. Consider an implementation of an unsorted singly linked list. Suppose it has its representation with a head pointer only. Given the representation, which of the following operations can be implemented in O(1) time?

i) Insertion at the front of the linked list

ii) Insertion at the end of the linked list

iii) Deletion of the front node of the linked list

iv) Deletion of the last node of the linked list

Answer

I and III

Status : Correct ***Marks : 1/1***

7. The following function takes a singly linked list of integers as a parameter and rearranges the elements of the lists.

The function is called with the list containing the integers 1, 2, 3, 4, 5, 6, 7 in the given order. What will be the contents of the list after the function completes execution?

```
struct node {
   int value;
   struct node* next;
};

void rearrange (struct node* list) {
   struct node *p,q;
   int temp;
   if (! List || ! list->next) return;
   p=list; q=list->next;
   while(q) {
      temp=p->value; p->value=q->value;
      q->value=temp;p=q->next;
      q=p?p->next:0;
   }
}
```

Answer

Status : Skipped ***Marks : 0/1***

8. Linked lists are not suitable for the implementation of?

Answer

Binary search

Status : Correct ***Marks : 1/1***

9. The following function reverse() is supposed to reverse a singly linked list. There is one line missing at the end of the function.

What should be added in place of "/*ADD A STATEMENT HERE*/", so that the function correctly reverses a linked list?

```
struct node {
   int data;
   struct node* next;
};
static void reverse(struct node** head_ref) {
   struct node* prev   = NULL;
   struct node* current = *head_ref;
   struct node* next;
   while (current != NULL) {
      next  = current->next;
      current->next = prev;
      prev = current;
      current = next;
   }
   /*ADD A STATEMENT HERE*/
}
```

Answer

*head_ref = prev;

Status : Correct ***Marks : 1/1***

10. In a singly linked list, what is the role of the "tail" node?

Answer

It stores the last element of the list

Status : Correct ***Marks : 1/1***

241901009

Rajalakshmi Engineering College

Name: Arya M R
Email: 241901009@rajalakshmi.edu.in
Roll no: 241901009
Phone: 7358633106
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 0_Arrays and Functions

Attempt : 3
Total Mark : 5
Marks Obtained : 4

Section 1 : Coding

1. Problem Statement

Alex, a budding programmer, is tasked with writing a menu-driven program to perform operations on an array of integers. The operations include finding the smallest number, the largest number, the sum of all numbers, and their average. The program must repeatedly display the menu until Alex chooses to exit.

Write a program to ensure the specified tasks are implemented based on Alex's choices.

Input Format

The first line contains an integer n, representing the number of elements in the array.

The second line contains n space-separated integers representing the array elements.

The subsequent lines contain integers representing the menu choices:

Choice 1: Find and display the smallest number.

Choice 2: Find and display the largest number.

Choice 3: Calculate and display the sum of all numbers.

Choice 4: Calculate and display the average of all numbers as double.

Choice 5: Exit the program.

Output Format

For each valid menu choice, print the corresponding result:

For choice 1, print "The smallest number is: X", where X is the smallest number in the array.

For choice 2, print "The largest number is: X", where X is the largest number in the array.

For choice 3, print "The sum of the numbers is: X", where X is the sum of all numbers in the array.

For choice 4, print "The average of the numbers is: X. XX", where X.XX is the double value representing an average of all numbers in the array, rounded to two decimal places.

For choice 5, print "Exiting the program".

If an invalid choice is made, print "Invalid choice! Please enter a valid option (1-5)."

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
10 20 30
1
5
Output: The smallest number is: 10
Exiting the program

Answer

```
#include <stdio.h>

int main() {
    int n, choice;

    // Input array size
    scanf("%d", &n);

    // Declare array with the given size
    int arr[n];

    // Input array elements
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Process menu choices until exit
    while (1) {
        scanf("%d", &choice);

        switch (choice) {
            case 1: { // Find smallest number
                int smallest = arr[0];
                for (int i = 1; i < n; i++) {
                    if (arr[i] < smallest) {
                        smallest = arr[i];
                    }
                }
                printf("The smallest number is: %d\n", smallest);
                break;
            }
            case 2: { // Find largest number
                int largest = arr[0];
                for (int i = 1; i < n; i++) {
```

```
            if (arr[i] > largest) {
                largest = arr[i];
            }
        }
        printf("The largest number is: %d\n", largest);
        break;
    }
    case 3: { // Calculate sum
        int sum = 0;
        for (int i = 0; i < n; i++) {
            sum += arr[i];
        }
        printf("The sum of the numbers is: %d\n", sum);
        break;
    }
    case 4: { // Calculate average
        int sum = 0;
        for (int i = 0; i < n; i++) {
            sum += arr[i];
        }
        double average = (double)sum / n;
        printf("The average of the numbers is: %.2f\n", average);
        break;
    }
    case 5: { // Exit
        printf("Exiting the program\n");
        return 0;
    }
    default: { // Invalid choice
        printf("Invalid choice! Please enter a valid option (1-5).\n");
        break;
    }
  }
 }

 return 0;
}
```

Status : Correct **Marks : 1/1**

2. Problem Statement

Saurabh is the manager of a growing tech company. He needs a program to record and analyze the monthly salaries of his employees. The program will take the number of employees and their respective salaries as input and then calculate the average salary, and find the highest and lowest salary among them.

Help Saurabh automate this task efficiently.

Input Format

The first line of input consists of an integer n, representing the number of employees.

The second line consists of n integers, where each integer represents the salary of an employee.

Output Format

The output prints n lines, where each line will display: "Employee i: "Salary

Where i is the employee number (starting from 1) and salary is the respective salary of that employee.

After that, print the average salary in the following format: "Average Salary: "average_salary

Where average_salary is the average salary of all employees, rounded to two decimal places.

Next, print the highest salary in the following format: "Highest Salary: "max_salary

Where max_salary is the highest salary among all employees.

241901009

Finally, print the lowest salary in the following format:"Lowest Salary: "min_salary

Where min_salary is the lowest salary among all employees.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
4000
3500
6000
2500
4500

Output: Employee 1: 4000
Employee 2: 3500
Employee 3: 6000
Employee 4: 2500
Employee 5: 4500

Average Salary: 4100.00
Highest Salary: 6000
Lowest Salary: 2500

Answer

```
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);

    int salaries[n];
    int highest_salary = 0;
    int lowest_salary = 1000000;
    long long total_salary = 0;
    for(int i = 0; i < n; i++) {
        scanf("%d", &salaries[i]);
        if(salaries[i] > highest_salary) {
            highest_salary = salaries[i];
```

```
    }
    if(salaries[i] < lowest_salary) {
        lowest_salary = salaries[i];
    }
    total_salary += salaries[i];
}
for(int i = 0; i < n; i++) {
    printf("Employee %d: %d\n", i+1, salaries[i]);
}
double average_salary = (double)total_salary / n;
printf("Average Salary: %.2f\n", average_salary);
printf("Highest Salary: %d\n", highest_salary);
printf("Lowest Salary: %d\n", lowest_salary);

return 0;
}
```

Status : Correct **Marks : 1/1**

3. Problem Statement

Write a program that will read a Matrix (two-dimensional arrays) and print the sum of all elements of each row by passing the matrix to a function.

Function Signature: void calculateRowSum(int [ ][ ], int, int)

Input Format

The first line consists of an integer M representing the number of rows.

The second line consists of an integer N representing the number of columns.

The next M lines consist of N space-separated integers in each line representing the elements of the matrix.

Output Format

The output displays the sum of all elements of each row separated by a space.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
3
1 2 3
4 5 6
7 8 9

Output: 6 15 24

Answer

```
#include <stdio.h>

void calculateRowSum(int matrix[][20], int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        int sum = 0;
        for (int j = 0; j < cols; j++) {
            sum += matrix[i][j];
        }

        printf("%d", sum);
        if (i < rows - 1) {
            printf(" ");
        }
    }
}

int main() {
    int matrix[20][20];
    int r, c;

    scanf("%d", &r);
    scanf("%d", &c);

    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }

    calculateRowSum(matrix, r, c);
    return 0;
```

}

Status : Correct ***Marks : 1/1***

4. Problem Statement

Write a program that reads an integer 'n' and a square matrix of size 'n x n' from the user. The program should then set all the elements in the lower triangular part of the matrix (including the main diagonal) to zero using a function and display the resulting matrix.

Function Signature: void setZeros(int [ ][ ], int)

Input Format

The first line consists of an integer M representing the number of rows & columns.

The next M lines consist of M space-separated integers in each line representing the elements of the matrix.

Output Format

The output displays the matrix containing M space-separated elements in M lines where the lower triangular elements are replaced with zero.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
10 20 30
40 50 60
70 80 90

Output: 0 20 30
0 0 60
0 0 0

Answer

```
#include <stdio.h>

void setZeros(int matrix[][10], int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j <= i; j++) {
            matrix[i][j] = 0;
        }
    }
}

int main() {
    int arr1[10][10];
    int n;

    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &arr1[i][j]);
        }
    }

    setZeros(arr1, n);

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", arr1[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Status : Correct **Marks : 1/1**

5. Problem Statement

Tim is creating a program to track and analyze student attendance. The program requires two inputs: the total number of students (n) and the total

number of class sessions (m). The task is to design and populate an attendance matrix, 'matrix', representing the attendance record of each student for each session.

The program's specific objective is to determine whether the last student on the list attended an even or odd number of classes. This functionality will aid teachers in quickly evaluating the attendance habits of individual students.

Input Format

The first line of input consists of a positive integer n, representing the number of students.

The second line consists of a positive integer m, representing the number of class sessions.

The next n lines consist of m space-separated positive integers representing the number of classes attended by the student.

Output Format

The output displays one of the following results:

If the last session is even the output prints "[LastSession] is even".

If the last session is odd the output prints "[LastSession] is odd".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 2
2
1 2
3 100
Output: 100 is even

Answer

```
#include <stdio.h>
```

```
int main() {
    int n, m;
    int matrix[n][m];
    scanf("%d", &n);
    scanf("%d", &m);
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }
    int lastSession = matrix[n-1][m-1];
    if (lastSession % 2 == 0) {
        printf("%d is even", lastSession);
    } else {
        printf("%d is odd", lastSession);
    }

    return 0;
}
```

Status : Wrong **Marks : 0/1**

241901009